Fuzzy Inference Systems for Search Result Quality Assessment

I. Introduction

Fuzzy logic is a form of many-valued logic that employs the concept of partial truth. processes that attributes multiple possible truth values to a single variable [1]. In contrast to Boolean logic where truth values are either 0 or 1, fuzzy truth values can be any real number between 0 and 1, allowing for the conceptual existence of partial truths. Dealing with large amounts of multivariate, sometimes arbitrary data brings increased levels of uncertainties, and fuzzy logic excels in this making sense of these.

Fuzzy logic's uses extend into data mining, business, forecasting, and many other decision-making fields [2]. Examples include medical fields to aid in diagnoses where uncertainty in data is common, financial sectors that assess investment risk, and image processing to handle uncertainty in pattern matching/object recognition. The main application implemented in this study is that of string matching. String matching allows us to calculate differences in lexical patterns, and measure their similarity in clean, understandable values. This is commonly seen on web searching platforms such as Google. Google provides implements spell checking mechanisms into their search feature that correct mistyped queries. If you were to type an incorrectly spelled word into the Google search bar, the results with which you are presented will be for the closest correctly spelled string to your input string. Functionally, this allows users to make more accurate queries, and it is the technology upon which I seek to improve.

II. Research

In my originally proposed idea, I aimed to implement a new type of string-matching method to an existing string matching system to improve web shopping results. I believed that by not only matching similarly spelled strings to an input string, but similarly connotated strings as well, a query for a certain product would become easier to complete.

Due to colloquial and dialectical differences between nations and intranational regions, queries of similar items could differ in the verbiage used to define said item. An example of this would be the regional differences surrounding soda in the United States, where some regions refer to any soda as "soda", some, as "pop", some as "cola", and some as "coke". These differences in language are easily missed by a simple pattern-matching algorithm, whose purpose is to evaluate lexical patterns instead of language connotations.

I tested this method on an existing fuzzy string-matching software called, TheFuzz, that uses Levenshtein Distance to calculate the differences between sequences [4]. TheFuzz utilizes some more certain pattern matching techniques from a program named RapidFuzz which expanded the TheFuzz program from Python to C++, and improved efficiency in the process.

Upon testing, the string-matching program performed as expected: it matched patterns with efficiency with multiple functions for partial and total matches of different kinds. The goal to build on top of this framework proved to be challenging with the time and human resource restrictions. The original scope of the project had to be scaled down as to be more manageable for a single person's capabilities.

Upon further research and deliberation, it was realized that creating a library for natural language processing has been implemented in many a search engine across the greater internet. To find an area of fuzzy string matching upon which I could improve, I shifted my focus to fine-tuning a very specific aspect of online marketplaces such as amazon.com.

I adjusted the project to attempt to perfect a querying model which displays items in a digital marketplace in the most relevant order by utilizing a rating system commonly found on shopping websites. Amazon.com has such a feature which is prominent next to every product profile: the star rating system.

I aimed to use aggregate customer rating data to evaluate the overall quality of a search result in a digital marketplace instead of simply displaying the nearest string-matched objects in order.

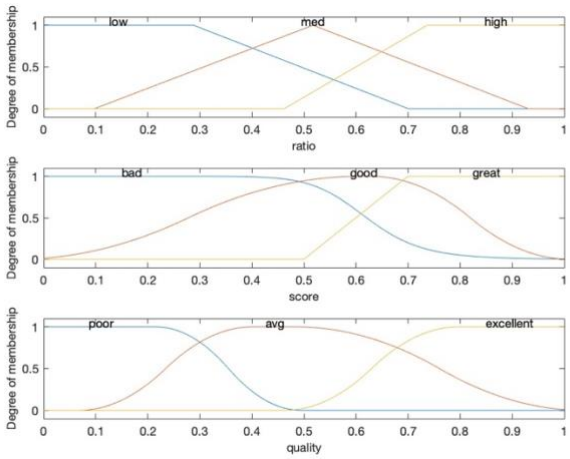
III. Experimentation

To achieve the desired outcome, I created a new Fuzzy Inference System to work in conjunction with TheFuzz, program that handles purely string matching.

TheFuzz examines two strings, one from user input and one from a dataset, and returns a clean 'ratio' of similarity on a [0, 1] scale. Strings that match better with each other return higher 'ratio' values. I would be utilizing this data to process through my FIS and return a new "quality" measure, which considers user rating on a data entry.

I created a Mamdani Type 1 Fuzzy Inference System where the defuzzification method is and quality and calculated using the centroid.

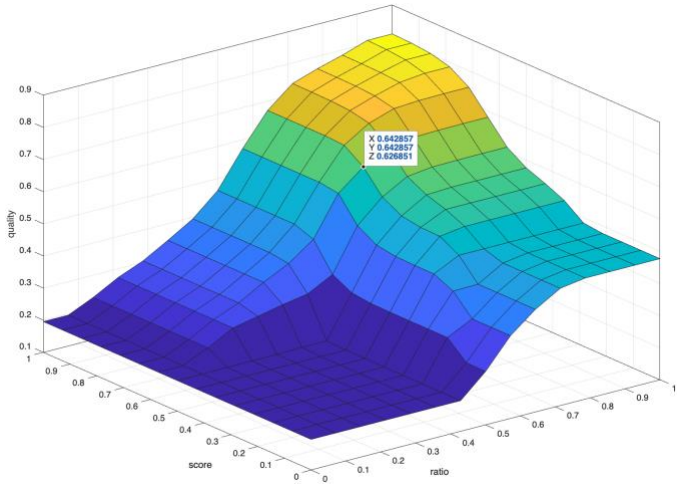
I created my own membership functions to that would accept ‘ratio’ values, and a ‘score’ which represents an aggregate user rating (*Figure 1).*



[Figure 1]

After creating the basic structure, I added some weighted rules to the system which would help shape the outputs in the most desirable manner.

I created a Control Surface diagram to better visualize how each entry into the program would score based on its fuzzified values (*Figure 2*).



[Figure 2]

Before beginning testing, I hypothesized that the results would favor data objects which have higher ‘score’ values, which would be evidence that my FIS was indeed impacting the results. Figures 3 and 4 show the results of testing (*Figure 3, Figure 4).*

```
Results:

disp([inputNames, outputName]);

    {'ratio'}    {'score'}    {'quality'}

disp([num2cell(testData), num2cell(output)]);

    {[0.7700]}    {[0.3200]}    {[0.5069]}
    {[0.6400]}    {[0.9000]}    {[0.6685]}
```

[Figure 3]

```
Results:

disp([inputNames, outputName]);

    {'ratio'}    {'score'}    {'quality'}

disp([num2cell(testData2), num2cell(output)]);

    {[0.7700]}    {[0.9400]}    {[0.7566]}
    {[     1]}    {[0.6500]}    {[0.6884]}
```

[Figure 4]

Objects with a higher ‘ratio’ which is the string-matched value supplied by TheFuzz score lower in ‘quality’ when their user ‘score’ is significantly lower. In both instances displayed in Figure 3 and Figure 4. In the practical case, the object with a higher ‘quality’ would be shown first in the query results.

IV. Reflection & Analysis

I feel that the FIS I created did a very good job as a general-purpose search result quality assessor. On one end of the spectrum, an object the is perfectly string matched to a user query will not be completely disregarded because of its user ‘score.’ There could be many reasons for an item to have a low user rating. One example could simply be due to a lack of reviews. The aggregate score of an item with only a handful of reviews is not indicative of a lack of quality of the item.

Another way I feel the FIS handled the data well was in its handling of string-matched values. Loosely pattern-matched items with great user scores appear in the search results at around the same level of priority that more strongly pattern-matched items with average user scores do. This shows the emphasis on the user rating taking effect in the system.

This system would likely best be utilized in a web-based marketplace that sells very similar items throughout its catalog. There are certainly query functions that can be added to this program to improve its accuracy, but most of them do not require nearly as complex construction or implementation. Just as well, this system could be fine-tuned to handle queries in many different contexts for online shopping that deal with a wider variety of items in a database.

V. References

[1] “Fuzzy logic,” *Wikipedia*, Nov. 23, 2023. https://en.wikipedia.org/wiki/Fuzzy_logic#Applications (accessed Dec. 10, 2023).

[2] Z. Liu, J. C. R. Alcantud, K. Qin and L. Xiong, "The Soft Sets and Fuzzy Sets-Based Neural Networks and Application," in IEEE Access, vol. 8, pp. 41615-41625, 2020, doi: 10.1109/ACCESS.2020.2976731.

[3] “Fuzzy Logic - Applications - Tutorialspoint,” *Tutorialspoint.com*, 2019. https://www.tutorialspoint.com/fuzzy_logic/fuzzy_logic_applications.htm

[4] “seatgeek/thefuzz,” *GitHub*, Mar. 07, 2022. https://github.com/seatgeek/thefuzz